Support Vector Machines: The Kernel Matters

Nathaniel Beckemeyer

1 Algorithm

For my Support Vector Machine project, I implemented the Sequential Minimal Optimization (SMO) algorithm provided in the CS 229 supplementary notes. See Table 1 for the parameters that I selected.

Tab. 1: The parameters used in my SMO implementation.

Parameter	Value
C	1E−10
tol	1E−7
max passes	10

The premise of the Support Vector Machine is to construct a decision boundary that maximizes the distance to the closest point on each side of the boundary, or the margin, through formulation as an optimization problem. The SMO algorithm iteratively attempts to solve this problem, with some modification to allow for error in constructing the boundary (which may handle outliers better).

Important is the idea of a kernel, which maps points from the space that they are provided in to some arbitrary space that the user selects; because the SMO algorithm only calculates inner products between vectors, if the kernel can be used to replace the inner product, then the mapping of features does not have to happen on each data vector, and can instead occur of each inner product.

2 Datasets

I ran the SMO algorithm with two kernels on six datasets. I used the Gaussian kernel, $K(x, z) = e^{-\frac{\|x-z\|^2}{2\sigma^2}}$, and a quintic kernel, $K(x, z) = (x^T z + 1)^5$. One of the datasets was the two spirals dataset; the other five were taken from the UCI Machine Learning Repository: The pop failures, diabetes, diagnostic, banknote authentication, and SPECTF datasets. The number of points and the number of features in each point can be seen in Table 2

Tab. 2: The parameters for each dataset.

Dataset	Examples	Features
Pop Failures	540	18
Diabetes	768	8
Diagnostic	569	30
Banknote Authentication	1372	4
SPECTF	187	45
Two Spirals	194	2

3 Experiments

For my experiments, I ran 50 trials for each dataset. Each trial, where m represents the total number of data points the data was broken into three sets: A test set of size $\lfloor \frac{m}{5} \rfloor$, a validation set of size $\lfloor \frac{m}{10} \rfloor$, and a training set containing the rest of the data points. The algorithm was trained on the training set, and its accuracy on the test set was recorded. Note that I actually did not use the validation set at all.

4 Results

Results are presented in Table 3.

Unsurprisingly, the quintic polynomial did not terminate for a single trial in several hours of running on the two spirals dataset; drawing a spiral with a quintic polynomial is impossible, and the parameters of my SMO implementation meant that misclassifications during training were virtually intolerable.

Notably, the Gaussian kernel outperformed the quintic kernel on every dataset at the $p < 0.0001$ significance level, measured by a two-tailed t-test, except for the diagnostic and pop failures datasets (where the differences are not statistically significant, even at much larger p values).

Tab. 3: The classification accuracy means and standard deviations for the SMO. All values provided are percents. Note that the italicized mean indicates that the classification accuracy listed is the complement of the accuracy generatd by the algorithm.

	Gaussian		Quintic	
Dataset	μ	σ	μ	σ
Pop Failures	94.76	2.09	94.26	1.80
Diabetes	65.19	4.03	52.41	12.60
Diagnostic	62.74	4.29	63.27	22.87
Banknote Authentication	100	0	99.60	0.46
SPECTF	97.57	4.10	*60.6*	39.44
Two Spirals	74.05	7.61	-	-

5 Discussions

I suspect that the performance advantage of the Gaussian kernel over the quintic kernel was caused more by overfitting of the quintic polynomials than by a failure to draw decision boundaries in the training set (due to the distribution of the dataset), for the following three reasons:

1. The relatively small number of examples in each dataset allowed overfitting to occur more readily.
2. The parameters of my SMO implementation did not well-tolerate misclassifcations during training, so overfitting may have been preferred in some cases.
3. The degree of the quintic kernel is 5—which is pretty high for usage with SVMs, and readily overfits data that is lower-dimensional.

The really curious cases are the ones where the Gaussian kernel performs no (statistically significantly) differently from the quintic kernel: I wonder what distributions and structures in the data led to this result. (Of course, this makes me wonder about distributions of data points in a more general sense, as well.)

Finding datasets in which the algorithm would terminate quickly for even one of the kernels proved to be rather challenging, which is understandable given the parameters that I selected for SMO.

6 Conclusion

I learned a lot about classification and the distribution of data (i.e., which spaces they are in or should be mapped into), thanks to the magic of kernels. I also gained a much greater understanding of the effect of the SMO parameters in how they relate to training times and overfitting.

Perhaps most importantly, I learned that SVMs are excellent off-the-shelf tools for classification; for instance, the SPECTF dataset with 45 features was classified with approximately 98% accuracy, and it took less than 40 seconds to run all 50 trials for the program—all I had to do was tell the program to use the Gaussian kernel.